COMPARISON OF CLUSTERING ALGORITHMS IN SPEAKER IDENTIFICATION

TOMI KINNUNEN, TEEMU KILPELÄINEN
and PASI FRÄNTI
{tkinnu, tkilpela, franti}@cs.joensuu.fi
Department of Computer Science, University of Joensuu,
Joensuu, FINLAND.

ABSTRACT

In speaker identification, we match a given (unkown) speaker to the set of known speakers in a database. The database is constructed from the speech samples of each known speaker. Feature vectors are extracted from the samples by short-term spectral analysis, and processed further by vector quantization for locating the clusters in the feature space. We study the role of the vector quantization in the speaker identification system. We compare the performance of different clustering algorithms, and the influence of the codebook size. We want to find out, which method provides the best clustering result, and whether the difference in quality contribute to improvement in recognition accuracy of the system.

Keywords: *Speech processing, speaker identification, vector quantization, clustering.*

1 INTRODUCTION

Speaker recognition is a generic term used for two related problems: speaker *identification* and *verification* [9]. In the identification task the goal is to recognize the unknown speaker from a set of *N* known speakers. In verification, an identity claim (e.g., a username) is given to the recognizer and the goal is to accept or reject the given identity claim. In this work we concentrate on the identification task.

The input of a speaker identification system is a sampled speech data, and the output is the index of the identified speaker. There are three important components in a speaker recognition system: the feature extraction component, the speaker models and the matching algorithm. Feature extractor derives a set of speaker-specific vectors from the input signal. Speaker model is then generated from these vectors for each speaker. The matching procedure performs the comparison of the speaker models. It is expected that the feature extraction is the most critical component of the system but it is also much more difficult part to be designed than the matching procedure.

In this work, we study the role of the *vector quantization* in a VQ-based speaker identification system [13]. We aim at solving this subproblem and give an answer to the question of which clustering algorithm we should use, and how large codebooks should be used. If we manage to do this, then we could fix this part of the algorithm and concentrate on more important subproblems of the system in the future.

We study the performance of several clustering algorithms, including three well known methods: *LBG*, *PNN*, and *self-organizing maps* (*SOM*), and few newer methods such as *iterative splitting* and *randomized local search* (*RLS*). We want to find out, which methods provide the best clustering results, and whether the difference in quality contributes to an improvement in the recognition accuracy of the identification system.

2 SPEAKER IDENTIFICATION SYSTEM

The structure of a VQ-based speaker identification system is illustrated in Fig. 1. There are two phases in the speaker identification: *training* and *recognition*. In the training phase, a mathematical model (VQ codebook in our case) is constructed for each speaker from their speech samples and the models are stored in the database. In recognition phase, the speech data of an unknown speakers is analyzed and the best matching model is searched from the database.

The analysis of the speech signals is based on short-term spectral analysis. The speech signal is decomposed into short fixed-length speech frames, which form the *feature vectors*. The feature extraction process is described more detailed in the Section 3.

The extracted feature vectors are processed further by vector quantization for locating the clusters in the feature space and for reducing the amount of data. The input of vector quantization is the set of feature vectors X and the output is a *codebook* C that consists of the cluster centroids, denoted as *code vectors*. The codebook represents the speaker model by approximating the distribution of the feature vectors in the feature space.

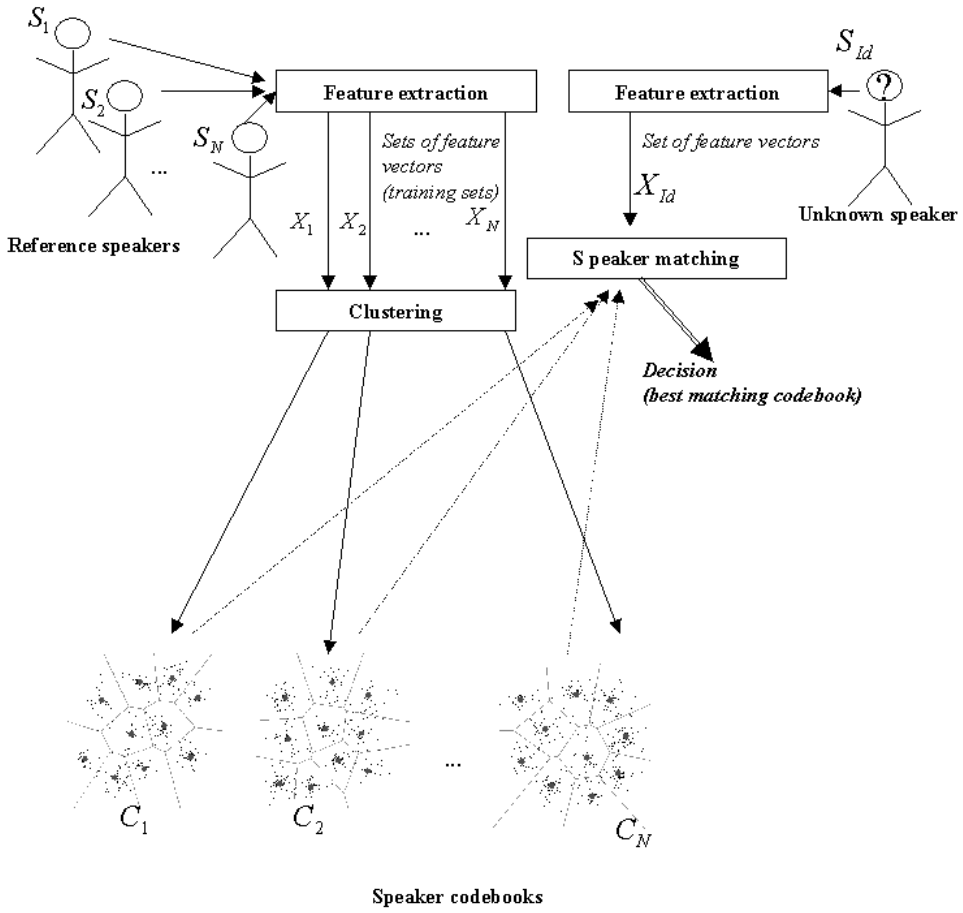

Fig. 1: *Structure of the VQ-based speaker identification system.*

The identification procedure is formulated as follows:

1. Compute the set of feature vectors $X = \{ \boldsymbol{x}_i \}$
2. **FOR EACH** speaker model C_i **DO**

   Compute the distortion $D_i = d(X, C_i)$ between X and C_i.
3. Identify the index of the unknown speaker *Id* as the one with the smallest distortion, i.e.

$$Id = \underset{i=1,...,N}{\operatorname{argmin}} \{D_i\}. \qquad (2.1)$$

The *distortion measure d* in the second step approximates the *dissimilarity* between the codebook $C_i = \{\boldsymbol{c}_{i1}, \boldsymbol{c}_{i2}, ..., \boldsymbol{c}_{iK}\}$ and the vector set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_L\}$. We use the most intuitive distortion measure; map each vector in X to the nearest code vector in C_i and compute the average of these distances:

$$d(X, C_i) = \frac{1}{L} \sum_{j=1}^{L} \min_{k=1}^{K} d_E(\boldsymbol{x}_j, \boldsymbol{c}_{ik}), \qquad (2.2)$$

where d_E is the Euclidean metric:

$$d_E(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{\dim} (\boldsymbol{x}_i - \boldsymbol{y}_i)^2} \qquad (2.3)$$

The distortion measure (2.2), known as the mean square error (*MSE*), gives also a measure for the quality of the codebook constructed from the training set X.

Note that in training phase we generate codebooks for the speakers, but in the recognition phase we perform a direct comparison between the set of feature vectors the codebooks to the codebooks of the known speakers. This arises the question whether we need the codebooks at the first place.

There are two good reasons for this: memory and time requirements. Computational load of the identification process becomes too high if we do not reduce the amount of data. It is very important to remove this kind of bottlenecks from a real-time speaker identification system. Memory consumption could also be a restricting factor in case of very large databases.

We assume that the feature vectors discriminate well the different acoustical units in the speech signal; similar phonemes (vectors) are located near to each other in the

feature space while different phonemes are far away from each other. When we perform the clustering of the feature vectors, we obtain efficient mean values of these different short-term acoustical units. The codebooks of different speakers may have some vectors very close to each other, but it is expected that there are enough dissimilar vectors so that the matching process can differentiate between codebooks of different speakers.

3 FEATURE EXTRACTION

Next, we describe the procedure for computing the feature vectors $c(n)$ from a given speech signal $s(n)$. The most commonly used features in speaker recognition systems are the features derived from the *cepstrum* [1]. Furui [8] was the first who applied cepstral analysis in speaker recognition.

Pre-emphasis

The speech is processed by a high-emphasis filter before input to the cepstrum analysis. This is due to the well-known fact that the higher frequencies contain more speaker-dependent information than the lower frequencies. We use a high pass filter whose transfer function is

$$H(z) = 1 - az^{-1}. \qquad (2.4)$$

Framing

The analysis of a discrete-time speech signal is based on *short-term* spectral analyses. This means that the signal is first divided into fixed-length short *frames*, e.g. 20 milliseconds. Adjacent frames usually overlap, e.g. by 10 milliseconds. After framing, these short-length "sub-signals" are considered as independent signals. For each frame, a fixed-length feature vector is computed, which describes the acoustic behavior of that particular frame.

Before frequency analysis, we apply a *window function* to the frames. The most simple windowing function is the *rectangular window*, i.e. "no window at all". However, usually smoother functions are used, and the most common in speech processing is the *Hamming window*. Smoother functions are better than rectangular window because the latter has abrupt discontinuities in its endpoints, which is undesirable for the frequency analysis [2].

Speech production modelling

Speech production can be well modeled by the *source-filter model* introduced by Fant [4]. According to the model, speech waveform is a result of two independent components: the *source* signal produced by vocal folds and the vocal tract *filter* which emphasizes certain frequencies of the source signal according to how it is configured. To be more precise, let us denote excitation source sequence by $e(n)$ and vocal tract filter signal as $v(n)$. The resulting speech waveform is simply a convolution

$$s(n) = e(n) * v(n). \qquad (3.1)$$

In frequency domain this becomes to

$$S(\omega) = E(\omega)V(\omega). \qquad (3.2)$$

The cepstrum

Fundamental idea of the cepstrum computation in speaker recognition is to discard the source characteristics because they contain much less information about the speaker identity than the vocal tract characteristics. In practice, the exact extraction of these two nonlinearly mixed signals $e(n)$ and $v(n)$ is impossible, but the cepstrum gives a good approximation for the "slow" spectral variations, i.e. the *envelope structure* of the signal, which characterizes the behavior of the vocal tract. Basically cepstrum computation is a deconvolution operator, which decomposes the signal into its source and filter characteristics. For the details about the way the cepstrum is computed, see e.g. [2].

The result of the deconvolution is a sequence of *cepstral coefficients* $\{c_0, c_1, ..., c_{M-1}\}$, where M is the desired number of coefficients. Coefficient c_0 corresponds to the total energy of the frame and thus contains no speaker information. Usually c_0 is discarded or used for normalization. In the cepstral domain, we use term *liftering* to point out that we want to "lifter" out those coefficients that describe fast spectral variations, i.e. the harmonic structure. In cepstral vector, lower coefficients describe the envelope structure and higher coefficients the harmonic structure [2].

3 VECTOR QUANTIZATION OF THE FEATURE VECTORS

There are two important design questions in vector quantization: the method for generating the codebook, and the size of the codebook. Next, we study known clustering algorithms for codebook generation. The question about the codebook size is issued in Section 4.

The clustering problem is defined as follows. Given a set of feature vectors $X = \{ \boldsymbol{x}_i \mid i = 1,...,L \}$, partition the data set into $K << L$ clusters such that similar vectors are grouped together and vectors with different features belong to different groups. The codebook $C = \{\boldsymbol{c}_1, ..., \boldsymbol{c}_K\}$ can then be constructed from the cluster representatives, which are the vector averages of each cluster.

We consider the following clustering algorithms:

- *Random*: Random codebook,
- *GLA*: Generalized Lloyd algorithm [11],
- *SOM*: Self-organizing maps [12],

- *PNN*: Pairwise nearest neighbor [3],
- *Split*: Iterative splitting technique [5],
- *RLS*: Randomized local search [7]

Random: A random codebook can be generated by selecting K random feature vectors. It serves as a point of comparison.

GLA: Generalized Lloyd algorithm (also known as *LBG*) starts with an initial codebook, which is iteratively improved until a local minimum is reached. In the first step, each feature vector is mapped to the nearest code vector in the current codebook. In the second step, the code vectors are recalculated as the centroids of the new partitions. The algorithm is iterated as long as improvement is achieved.

SOM: Self-organizing maps is a neural network approach to the clustering. The neurons in the network are connected with a 1-D or 2-D structure, and they correspond to the codevectors. The feature vectors are feed to the network by finding the nearest codevector for each input vector. The best matched codevector and its neighboring vectors (according to the network structure) are updated by moving it towards the input vector. After processing the training set by a predefined number of times, the neighborhood size is shrunk and the entire process is repeated until the neighborhood shrinks to zero.

PNN: Pairwise nearest neighbor generates the codebook hierarchically. It starts by initializing each training vector as a separate code vector. Two code vectors are merged at each step of the algorithm and the process is repeated until the desired size of the codebook is obtained. The code vectors to be merged are always the ones whose merge increase the distortion least. We use the fast exact PNN method introduced in [6].

Split: An opposite, top-down approach starts with a single cluster including all the feature vectors. New clusters are then created one at a time by dividing existing clusters. The splitting process is repeated until the desired number of clusters is reached. The divisive approach usually requires much less computation than the PNN. The best known approach for the splitting is to use *principal component analysis* (PCA). This method gives comparable results to that of the PNN with much faster algorithm.

RLS: Randomized local search algorithm starts with a random codebook, which is then improved by a predefined number of iterations. At each step, a new candidate solution is generated using the following operations. The clustering structure of the current solution is first modified using so-called *random swap* technique, in which a randomly chosen code vector is replaced by another randomly chosen input vector. The partition of the new solution is then adjusted in respect to the modified codebook. Two iterations of the GLA are then applied to fine-tune the trial solution. The candidate is evaluated and accepted if it improves the previous solution. The algorithm is iterated for a fixed number of iterations.

4 EXPERIMENTAL RESULTS

We collected a speaker database of 25 speakers (14 males + 11 females). Speech was recorded in a laboratory environment with a PC computer. For each speaker we recorded two utterances of Finnish speech: one for training and the other for recognition. Every speaker read the same sentences. Summary of the speech database is given in Table 1.

Table 1: Summary of the speaker database.

# Speakers	25 (14 M + 11 F)
Avg. duration of training utterance	66.5 s
Avg. duration of recognition utterance	17.7 s
Sampling & quantization	11.025 kHz, 16 bits.

Before analysis, the speech files were anti-alias filtered and downsampled to 8.0 kHz. After that, silent parts were removed using short-term energy calculations. The feature extraction itself was performed as follows: remove DC offset, high-emphasis filtering with $H(z) = 1 - 0.95z^{-1}$, and, finally, perform short-term mel-cepstrum analysis with a 30 ms Hamming-window, with 10 ms shift. The number of mel-cepstral coefficients (dimension of feature vectors) were selected as 12. As usually, coefficient c_0 was discarded.

We evaluated the performance of the five different clustering algorithms of Section 3. As the measure of quality for a given VQ codebook, we use the mean squared error between the training set and the resulting codebook C_i. The resulting MSE-values are shown in Fig. 2 and 3 with two different sizes of codebook (K=64 and K=256).

The results show that there are only a small difference between the best clustering algorithms. Even the standard GLA method gave MSE-values close to that of the best method, RLS. The corresponding identification rates are shown in Fig. 4 and 5. The choice of the clustering method have only a marginal effect on the identification rate.

The effect of the codebook size is illustrated in Fig. 6 and 7 for the best method (RLS) and for the random codebook. The identification rates clearly increases with respect to the codebook size. If it is set to 128 or higher, even with the random codebook the method is capable of identifying 96% of the speakers, which corresponds to a single miss-classification. With the best clustering methods (RLS, SPLIT), the identification rate does not improve anymore for codebooks of sizes > 64.

Finally, the running times for generating the codebooks are shown in Fig. 8. If the database can be constructed off-line, the running times are hardly significant. The RLS method takes slightly longer time than the rest of the methods because it was tuned for quality and not for speed. If the running time was critical, then the SPLIT method would be a good choice.

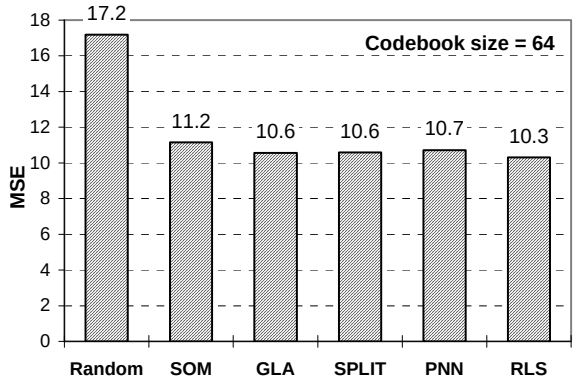

Fig. 2: *Quality of the codebook (scaled MSE-values) using different clustering algorithms. K=128.*

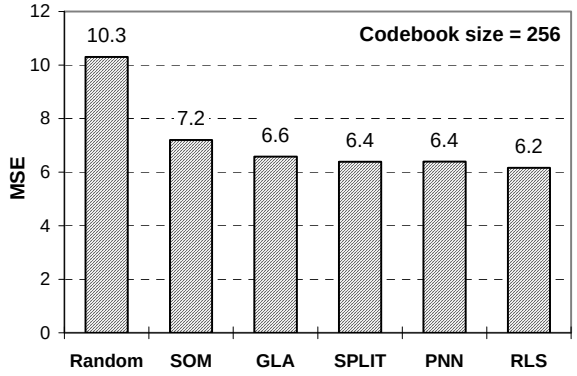

Fig. 3: *Quality of the codebook (scaled MSE-values) using different clustering algorithms. K=256.*

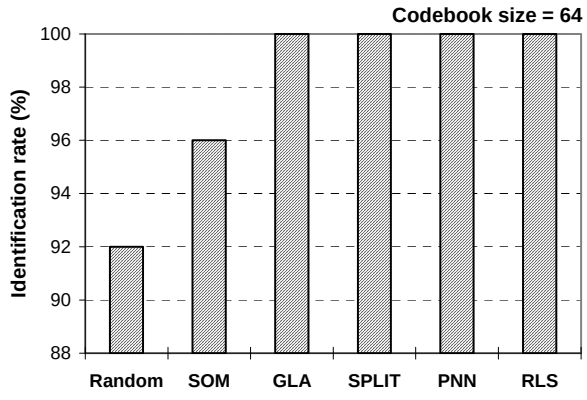

Fig. 4: *Identification accuracy of the algorithms. K=64.*

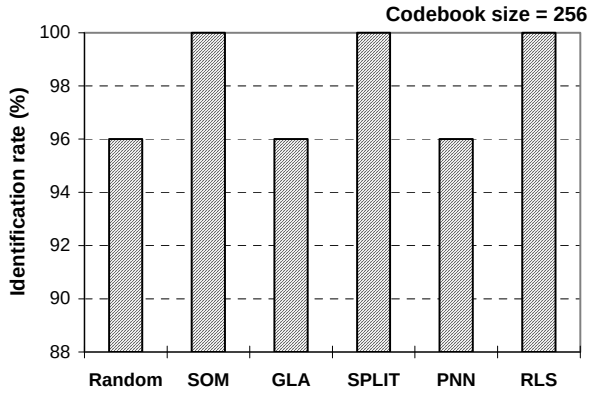

Fig. 5: *Identification accuracy of the algorithms. K=256.*

5 CONCLUSIONS

We evaluated the performance of five different clustering algorithms for VQ-based speaker identification. We noticed that the MSE-values of the codebooks produced by the algorithms were only marginally different, and the corresponding recognition rates were rather similar.

The easiest way for improving the identification accuracy was to increase the codebook size high enough. No side-effect was observed due to the increase, except the increase in the running. However, codebooks of size greater than 64 did not have any further impact as the identification rate already reached 100%.

We conclude that the fastest algorithm (SPLIT) should be used if the speaker database is very large and running time important. Otherwise, we recommend to use the best algorithm (RLS) because it is simpler to implement and, after all, it gave the best codebooks even though the difference was only marginal for our database.

It is noted that the speaker database was relatively small, the speech samples were quite long, and they were generated in laboratory conditions. Future experiments must therefore be made in more demanding environments in order to obtain more conclusive results.

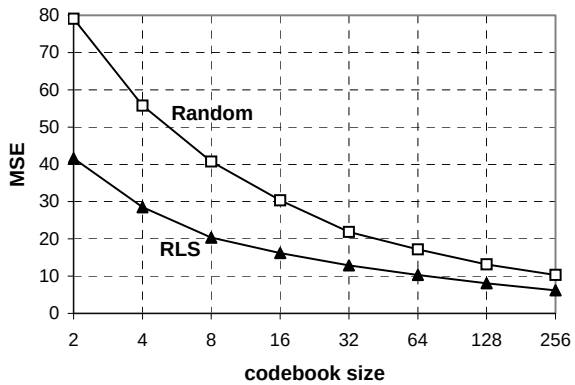

Fig. 6: *Quality of the codebook as a function of the codebook size.*

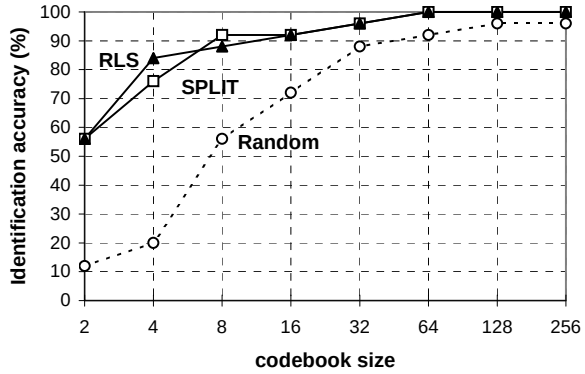

Fig. 7: *Identification accuracy of the algorithms as a function of the codebook size.*

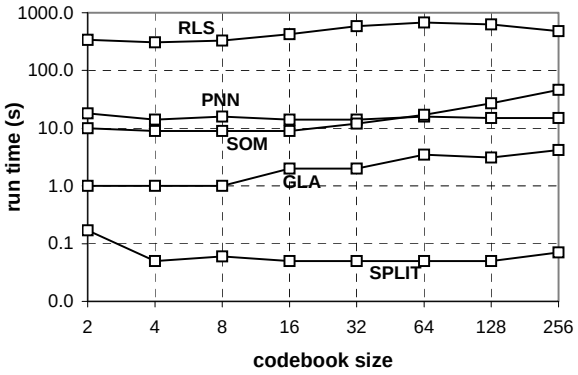

Fig. 8: *Run times of the clustering algorithms.*

REFERENCES

[1] Bogert B.P., Healy M.J.R., Tukey J.W.: ”The frequency alanysis of time series for echoes: cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking”, *Proc. Symposium Time Series Analysis*, John Wiley and Sons, NY, 209-243, 1963.

[2] Deller Jr. J.R., Proakis J.G., Hansen J.H.L.: *Discrete-Time Processing of Speech Signals*. (Macmillan Publishing Company, New York, 2000).

[3] W.H. Equitz, “A new vector quantization clustering algorithm”, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 37 (10): 1568-1575, October 1989.

[4] G. Fant: *Acoustic Theory of Speech Production*, (Mouton, The Hague, 1960).

[5] P. Fränti, T. Kaukoranta and O. Nevalainen, “On the splitting method for vector quantization codebook generation”, *Optical Engineering*, 36 (11): 3043-3051, November 1997.

[6] P. Fränti, T. Kaukoranta, D.-F. Shen and K.-S. Chang, “Fast and memory efficient implementation of the exact PNN”, *IEEE Trans. on Image Processing*, 9 (5): May 2000.

[7] P. Fränti and J. Kivijärvi, “Randomized local search algorithm for the clustering problem”, *Pattern Analysis and Applications*. (to appear)

[8] S. Furui: “Cepstral analysis technique for automatic speaker verification”. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 29(2): 254-272, 1981.

[9] S. Furui: ”Recent advances in speaker recognition”. *Pattern Recognition Letters*, 18: 859-872, 1997.

[10] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*. (Kluwer Academic Publishers, Dordrecht, 1992).

[11] Y. Linde, A. Buzo and R.M. Gray, “An algorithm for vector quantizer design”. *IEEE Trans. on Communications*, 28 (1): 84-95, January 1980.

[12] N.M. Nasrabadi and Y. Feng, "Vector quantization of images based upon the Kohonen self-organization feature maps", *Neural Networks*, 1: 518 (1988).

[13] F.K. Soong, A.E. Rosenberg, B-H. Juang, L.R. Rabiner: ”A vector quantization approach to speaker recognition”. *AT&T Technical Journal*, 66: 14-26, 1987.